# Lecture 1: Intro (skim) .

Attributes of intelligence:

* The ability to solve problems.
* The ability to tackle ambiguous & complex problems.
* The ability to learn & recognize things/patterns.
* The ability to understand and perceive.

Definition of intelligence by different people:

1. Lloyd G. Humphreys: the result of the process of acquiring, storing, retrieving, combining, comparing, and using information in new contexts/ conceptual skills.
2. McCarthy (on artificial intelligence):

* the science & engineering of making intelligent machines.
* The goal of AI is to develop machines that act as if they are intelligent.

1. Winston (on artificial intelligence): the study of computation that makes it possible to perceive, reason, and act.
2. Russel (on artificial intelligence): a system that perceives its environment & takes actions to maximize its chances of success.

Fields that advanced Ai:

1. Philosophy:

* Where does knowledge come from?
* How does knowledge lead to action?

1. Mathematics:

* What can be computed?
* How do we work with uncertain information?

1. Neuroscience:

* How does our brain process information?

1. Psychology.

* How do humans and animals think & act?

1. Economics.

* How should we make decisions to maximize payoff?

1. Computer Engineering.

* How can we build an efficient computer?
* How can systems operate under their own control?

1. Linguistics.

* How does language relate to thought?

History of Ai.

Turing test:

* Proposed by Alan Turing in 1950.
* The AI is given a question and a human is given the same question, then a judge tries to find out which answer belongs to the AI.

General Problem Solver (GPS):

* Proposed by Newell, Simon, and Shaw in 1959.
* Based on Polya’s problem solving heuristics:
  + Understand the problem.
  + Devise a plan.
  + Carry out the plan.
  + Look back at the solution.

MYCIN:

* One of the first rule-based expert systems.
* Published in 1969.

The Defense Advanced Research Projects Agency (DARPA):

* Created in 1983 by the US government.
* Funds AI projects.

Automated factories:

* In 1991, some factories, mostly producing cars, became fully automated.

Deep Blue:

* This is an AI system developed to play chess by IBM.
* In 1997, it defeated the world chess champion in a six-game match.

Stanford’s self-driven car:

* In 2005 Stanford university racing team created their own self-driving car.
* It was fully autonomous & they won 2 mil.
* It was called “Stanley”.

Table

Description automatically generatedProgramming w/wo AI.

Ai applications:

* Gaming:
  + AI is important in strategic games.
  + It can think of a large number of possible positions.
* Natural Language Processing:
  + AI that understands our natural languages.
* Expert Systems:
  + Systems that use software & hardware to advise/ provide an explanation about sth.
* Vision Systems:
  + Systems that understand, interpret, and comprehend visual inputs.
* Speech recognition:
  + Systems that can translate speech and comprehend it.
* Handwriting recognition:
  + Software used to extract text written on a paper/screen.
* Intelligent robots:
  + Robots that can perform various tasks.
  + They have sensors to detect most things.
  + They can learn from their mistakes & adapt.

Ai benefits:

* High accuracy with fewer errors.
* High speed.
* Less routine.
* Useful for risky areas.
* No human error.
* Can work as a Digital assistant.

The Chinese room argument.

It states that even if a computer program can produce output that appears to be indistinguishable from that of a human speaker of a language, it still does not truly understand the meaning of the language it is processing.

* Proposed by John Searle in 1980.
* Weak AI hypothesis: AI that simulates animals/humans.
* Strong AI Hypothesis: AI that is actually intelligent.

# Lecture 2: Knowledge representation.

Diagram

Description automatically generatedHumans are better than Ai at:

* Understanding.
* Reasoning.
* Interpreting knowledge.

Requirements for Ai to solve a problem:

* Knowledge:
  + The information related to the surrounding environment.
* Reasoning:
  + The ability to understand stored knowledge.
* Intelligence:
  + The ability to make decisions based on the stored information.

Problems with knowledge:

* Data volume is huge (big data).
* Not well organized/well formatted (unstructured).
* It keeps changing (dynamic).

AI Technique: a way to organize & use knowledge efficiently.

* It should be understandable by the providers.
* Should be easily modifiable.
* Should be useful in many situations.

## Intelligence.

Intelligence is intangible.

Its components:

1. Reasoning.
   1. Inductive reasoning: performs specific observations to make general statements.
   2. Deductive reasoning: starts with a general statement & examines the possibilities of reaching specific/logical conclusion.
2. Learning.

The activity of gaining knowledge/skill. It enhances the awareness of the AI.

Types of learning:

* + Auditory learning.
  + Motor learning.
  + Observational learning.
  + Spatial learning.
  + Stimulus-Response learning.

1. Problem solving.

The process of arriving at a desired solution by taking some path (may have some constraints).

Decision making: the process of choosing the best path to reach the goal.

1. Perception.

The process of acquiring, interpreting, selecting, and organizing sensory info.

Perception depends on sensors.

1. Linguistic intelligence.

the ability to use, comprehend, speak, and write verbal & written language.

## Knowledge representation (KR) language.

KR is the study of how beliefs, intentions, and judgement of an intelligent system can be expressed correctly for automated reasoning.

* KR2 (KR&R) is dedicated to representing information in a form that the computer can use to solve tasks.
  + Objects.
  + Properties.
  + Categories.
  + Relationships.
  + Situations.
  + Events.
* It isn’t just about storing data.
* It allows a machine to learn from that knowledge & behave intelligently.

KR has two aspects:

* The syntax: describes the possible configurations that can form sentences.
* The semantics: the facts in the world that the sentences refer to.

All reasoning mechanisms must operate on fact representation not the facts themselves.

Reasoning: the process of constructing new physical configurations from old ones.

* The new sentence must be true given that the old sentences are true.
  + This is called Entailment (KB |= A).
  + Based on semantics.

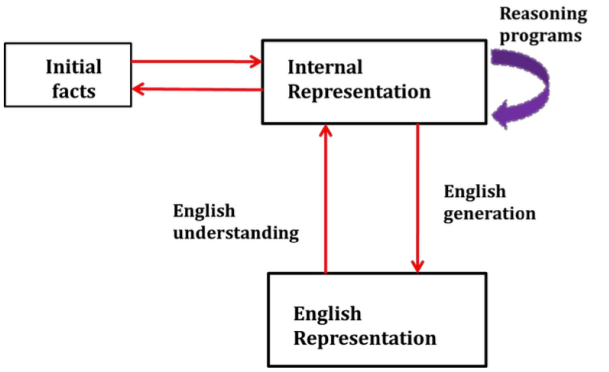
Inference procedure actions:

* Deduction: given a Knowledge Base (KB), it can generate new sentences.
* Entailment: a relationship between sentences based on semantics.

# Lecture 3: KR (cont’d).

* Humans are good at understanding, reasoning, and interpreting knowledge.
* Humans store & recall information by patterns.
  + Machines do it by searching algorithms.
* Humans can imagine how an object looks like even if some part is missing.
  + Machines can’t do it correctly.
* AI researchers are concerned with knowledge representation & search algorithms.

## Representation & mapping.

* Mapping happens between facts & representations.
* Computers can’t understand natural language (English) and can’t draw inferences from it.

## Representation schemes:

* Logical representation.
* Procedural representation.
* Network representation.
* Structured representation.

### Scheme judgement:

* Expressiveness.
  + can it clearly & completely represent necessary data?
  + Is it able to handle different types, levels & structures, relationships of knowledge.
  + Related to representational adequacy property.
* Effectiveness.
  + Can we use the data for computation & inference?
  + Must provide a way to infer new knowledge from old ones.
  + Should be amenable to computation.
  + Related to inferential adequacy property.
* Efficiency.
  + Can we gather & harness data easily?
  + New knowledge must be generated efficiently.
  + Related to inferential efficiency & acquisitional efficiency property.
* Explicitness.
  + Does it provide clear explanations?
  + Should provide an explanation for its new knowledge.
  + Should allow justifications for its reasoning.
  + Related to inferential adequacy property.

### Scheme parts:

* Lexical part: describes allowable vocabulary.
* Structural part: describes the constraints on combinations.
* Semantic part: associates a meaning with a description.
* Procedural part: describes access procedures (constructors).

## Semantic networks.

* A node & link knowledge description technique.

Its parts:

* Lexical.
  + Nodes: represent objects (circles, ellipses, rectangles).
  + Links: connects nodes (arrow from one node to another).

Diagram

Description automatically generated

* Structural.
  + Each link connects the tail node to head node.
* Semantic.
  + Node links denote application-specific entities.
* Procedural.
  + Can construct a node || link.
  + Can create a list of nodes connected to each other.

### Semantic network advantages:

* Natural representation of knowledge.
* Simple & easily understood.
* Efficient.
* Translatable to prolog.

### Semantic network disadvantages:

* High computational time at runtime.
  + Need to traverse the complete network tree to answer a question.
* Doesn’t have any equivalent quantifier (for all, for some, none).
* Doesn’t have any standard definition for link names.
* Not intelligent & depends on the creator.
* Tries to build a human-like memory but can’t.

# Lecture 4: State space search.

Feasible solution: solution that satisfies all defined constraints & requirements.

* Can be rated by its value & fitness to the problem.

State space: space that contains all feasible solutions.

* Searching for a solution = searching for an extreme solution (max || min).

Optimal solution: sequence of actions that has the lowest path cost.

Diagram

Description automatically generated

## State space representation Components.

* Static state.
* A picture containing clock

  Description automatically generatedTransitions between states.

## Search problem components.

* Initial state.
* Actions.
* Transition model (result state of each action).
* Goal state(s).
* Solution path.
* Path cost.

## Formal state space.

Represented by NASG.

* Set of Nodes in the graph (states in the problem-solving process).
* Set of Arcs between nodes (steps in the problem-solving process).
* Set of Start state(s).
* Set of Goal state(s).
  + Described by a measurable property of the encountered states.

## Search strategies evaluation.

* Completeness (does it find the solution).
* Time complexity (number of nodes generated).
* Space complexity (max number of nodes in memory).
* Optimality (does it always find the least-cost solution).

## Blind vs Heuristic strategies.

Search strategy defined by order of node expansion.

* Blind (uninformed, exhaustive, brute-force).
  + Doesn’t exploit any of the state’s info.
* Heuristic (informed).
  + exploits info to assess which node is better.

## Search.

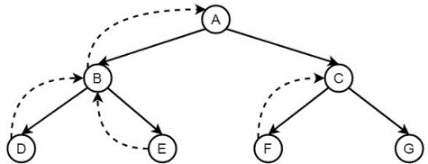
### Basic idea.

* Initialize frontier using initial state.
* Loop.
  + If frontier is empty.
    - Return failure (Stop).
  + Choose a leaf node & remove it from the frontier.
  + If current node = goal node.
    - Return the current solution.
  + Else.
    - Expand the current node & add its children to the frontier.

### Handling repeated states.

* Each time a node is expanded, add that node to Explored set.
  + Don’t add explored nodes in the frontier.
* Each time you add a node to the frontier.
  + Check if it already exists with higher path cost (yes => replace node with new one).

### Backtrack search.

* Technique that systematically tries all paths through a state space.
* Starts at the initial node & keeps track of the current path until:
  + It finds the goal node (stops & returns path).
  + It finds a dead end (backtracks to the most recent node).

### Breadth-first strategy (BFS).

* Uses a queue to represent the frontier.
* Searches row by row.
* Checks if the current node is the goal node.
  + If not, it adds its children to the queue.

Text

Description automatically generated

### Depth-first strategy (DFS).

* Uses a stack to represent the frontier.
* Checks if the current node is the goal node.
  + If not, it pushes its children to the stack and checks them first.

Text

Description automatically generated

### Other strategies.

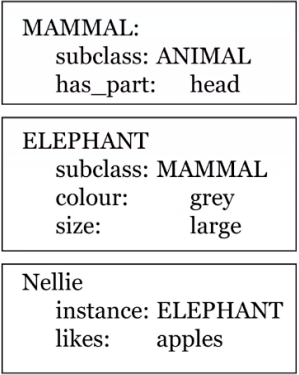
* Depth limited:
  + Uses depth-first but limits the depth (using var K).
  + Can sometimes return failure even if a solution exists.
* Iterative deepening:
  + Same as depth limited but K varies (starts at 1 & increases if the solution was not found).
  + Uses depth-first with depth cutoff K.

# Lecture 5: Representation.

Knowledge: collection of facts from some domain.

* For a program to manipulate these facts, a suitable representation is required.
* Good representation facilitates problem representation.
* Representation Methods:
  + Sematic networks.
  + Frames.
  + Predicate logic.
  + Production rules.

## Frames.

* A complex entity that contains all information relevant to a particular concept.
* OR a group of slots & fillers that define a stereotypical object.
* A complete frame consists of a whole hierarchy/network of frames connected by links/pointers.
* Frame Types:
  + Class frame.
  + Individual (instance frame).

### Frame Slots:

* Text

  Description automatically generated with medium confidenceA slot contains one side or more.
* Diagram

  Description automatically generated with low confidenceA side may contain:
  + Values/defaults.
  + Ranges.
  + Pointers (to other frames).
  + Set of rules/procedures.
    - If added: a procedural attachment that specifies an action in case a value is added/changed.
    - If needed: a procedural attachment that gets information when a slot is empty.

### Advantages of frames.

* Makes programming easier (groups related data).
* Inheritance is easily controlled.
* Adding slots, new attributes, and relations is easy.
* Default data & searching is easy.

### Disadvantages of frames.

* No standards for defining frame-based systems.
* Inference mechanism is not easily processed.
* Can’t represent exceptions.

## Semantic network to frame.

Diagram

Description automatically generated

## First-Order Logic (FOL) syntax.

* Constants.
  + name of a specific object (human, horse, number, etc.).
* Predicates.
  + State relation between objects (returns true or false based on inputs).
  + Ex. Married (X,Y).
  + Proposition: a predicate after it’s given a value.
    - Ex) Married(Ahmed, Mona) => true || false (based on data).
* Functions.
  + Maps from one object to another.
  + EX. Father(Hamada) => Ebrahim (Ebrahim is hamada’s father).
* Variables.
  + Refers to objects without names (X, Y, A, etc.).
* Connectivity:
  + Two-way relation ().
  + One-way relation ()
  + Or ().
  + And ().
  + Not ().
* Equality ().
* Quantifiers:
  + All ().
  + Some ().

# Lecture 6: production system.

Production systems (rule-based system): a system based on IF…Then rules.

Its parts:

* Production rules:
  + Represented in a pair consisting of a Condition & an action.
  + Represented as a single chunk of problem-solving knowledge.
  + The condition is a pattern that determines when the rule may be applied.
  + The action defines the associated problem-solving step(s).
* Working memory:
  + Contains a description of the current state.
  + Used for matching rules.
* Control structure:
  + Other names (interpreter || recognize-act cycle).
  + Implements search (moves the system towards the goal).

## Rule Characteristics:

* Condition:
  + Name: premise || antecedent || situation || IF.
  + Nature: conditions (like declarative knowledge).
  + Size: can have many IFs.
  + Statement: can contain AND/OR statements.
* Action:
  + Name: conclusion || consequence || action || then.
  + Nature: resolution (like procedural knowledge).
  + Size: usually one conclusion.
  + Statement:
    - AND: All must be true.
    - OR: one condition must be true.

## Rule Chaining:

Rule chain: a set of rules in a sequence, where every rule produces a fact used in another rule.

* A component in inference procedure (uses logical rules to get new info).
* Types:
  + Forward chaining.
  + Backward chaining.

### Forward chaining (Data-driven inference):

* Facts are represented in working memory & are constantly updated.
* Rules represent the possible actions to take when the specified conditions require facts present in working memory.
* Conflict set: all rules whose facts match the facts in the memory.
* Rule fired: the one rule from the conflict set that got executed.

Diagram

Description automatically generated

* Steps:
  + Start at the current state.
  + Match the arguments of the rules.
  + Perform the corresponding actions.
  + Update the working memory based on the actions.
  + Table

    Description automatically generatedProcess continues until:
    - no more rules can be applied.
    - cycle limit is met.

### Diagram Description automatically generatedBackward chaining (Goal-driven inference):

* works towards a final state.
* Looks at the memory to see if the sub-goal states already exist.
* The actions of those rules are identified.
* New sub-goals are set up for achieving the arguments of those rules.

### Forward or backward chaining.

* Forward:
  + If you need to gather info, then infer from it.
* Backward:
  + If you begin with a hypothetical solution, then attempt to find facts to prove it.

## Conflict resolution:

* Conflict happens when two or more rules can be fired at the same time.
* The inference engine must determine which rule to fire.
* Conflict resolution: a method for choosing which rule to fire.
* Ex)
  + R1: If the traffic light is red THEN go.
  + R2: If the traffic light is red THEN stop.

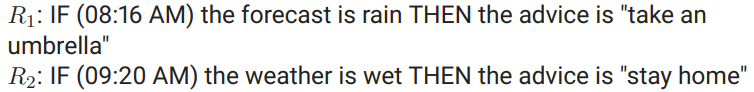
### Conflict resolution methods:

1. Icon

   Description automatically generated with low confidenceFiring the rule with the highest priority.
   1. Priority is established by placing rules in an appropriate order.



1. Fire the most specific rule (longest matching strategy).
   1. Assumes that that rule processes more information.
2. Fire the rule that uses the most recently entered data.
   1. Relies on time tags attached to each fact.
   2. Useful for real-time expert systems (data is always updating).



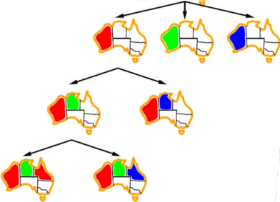
# Lecture 7: constraint Satisfaction Problems (CSP).

* Previous search problems use atomic states (no internal structure).
* CSPs are a special kind of search problems.
  + Uses a factored representation for the state space.
  + Each state has a set of variables (with values).
  + The Problem is solved when all variables have values that satisfies all constraints.
* CSP components:
  + X: set of variables (X1, X2, …, Xn).
  + D: set of domains (D1, D2, …, Dn).
  + C: set of constraints (specify allowable value combinations).
    - Implicit constraint: a constraint represented by a math formula or using FOL.
    - explicit constraint: constraint that lists all possible combinations.
* Tries to remove large parts of the search space (removes the ones that violate the constraints).
* State: defined by assigning values to some/all variables.
* Consistent assignment: assigning a value to a variable without violating any constraint.
* Complete assignment: assigning a value to each variable.
* Partial assignment: assigning a value to some variables.
* Solution: found when assignment is consistent and complete (satisfying all constraints).
  + Appears at depth n.

## CSP formulation:

* Initial state: all variables don’t have values.
* Goal state: the state where all variables have values.
* Path cost: a constant cost for every step.
  + The path itself is irrelevant.

## Backtracking with Forward Checking Strategy.

* Forward checking: crossing any values that violate a constraint (from the next assignment values).
* Increases search speeds.
* Ex. Adjacent cities must have different colors.

### Algorithm:

A picture containing text, screenshot, font

Description automatically generated

## Assignment ordering:

### Using degree of heuristics:

* A picture containing text, font, screenshot, design

  Description automatically generatedThe next assigned node is the one that has the maximum number of constraints with other nodes.
* Ex. in this picture the next node is B.

### Using the least-constraining value heuristics:

* The next assigned value is the value that leaves the most options in the domains of other nodes.